Progetto DSABD

Sommario

Indice delle figure

Indice delle Tabelle

I. Abstract

Il progetto consiste nello sviluppo di un sistema distribuito basato su microservizi per la gestione di utenti e dati finanziari. L'applicazione consente agli utenti di registrarsi, aggiornare, eliminare informazioni e recuperare i dati azionari attraverso l'integrazione con la libreria Yahoo Finance. Il sistema include un server grpc per la gestione delle richieste client, un Data Collector per l'acquisizione dei dati azionari yfinance e un database per l'acquisizione persistenti delle informazioni. Questo modello permette di garantire ad ogni utente di associare uno o più ticker, dei quali viene tenuta traccia dei suoi storici.

II. Schema Architetturale

Il sistema si basa su un'architettura a microservizi, in cui ogni componente è indipendente e responsabile di una parte specifica, ma interagiscono attraverso interfacce definite. Questa implementazione permette scalabilità, modularità. manutenibilità.

Sono definiti tre microservizi:

- **Server Microservice**: permette di gestire le richieste. Effettua in particolare le seguenti azioni:
  - **Gestione delle richieste del client:** le richieste vengono ricevute tramite il protocollo grpc permettendo una comunicazione bidirezionale.
  - **Interazione con il database:** il sever interagisce direttamente con il database per archiviare o modificare informazioni, così da fornire i dati richiesti al client.
  - **Gestione delle API:** il server gestisce le varie operazione con la chiamata di diverse API che consento ai client di interagire con il sistema.
- **Data Collector Microservice**: si occupa di recuperare periodicamente i dati associati ai titoli azionari, utilizzando la libreria yfinance. Effettua interazioni specifiche basate su:
  - **Recupero dati**: recupero elenco dei tickers per aggiornare successivamente i dati
  - **Inserimento dati**: inserisce i dati aggiornati periodicamente all'interno del database per garantire la persistenza dei dati e una successiva consultazione.
- **Database**: fondamentale per rendere i dati persistenti, utilizzando la politica per la gestione e archiviazione delle informazioni basate su MySQL. Effettua le seguenti operazioni:
  - **Archiviazion**e: permette di archiviare utenti, ticker e dati storici relativi ai titoli azionari includendo sia i valori che i timestamp associati.
  - **Relazione**: permette di stabilire la relazione tra i diversi dati come tra utenti e i tickers, consentendo di associare ad ogni utente il ticker d'interesse.

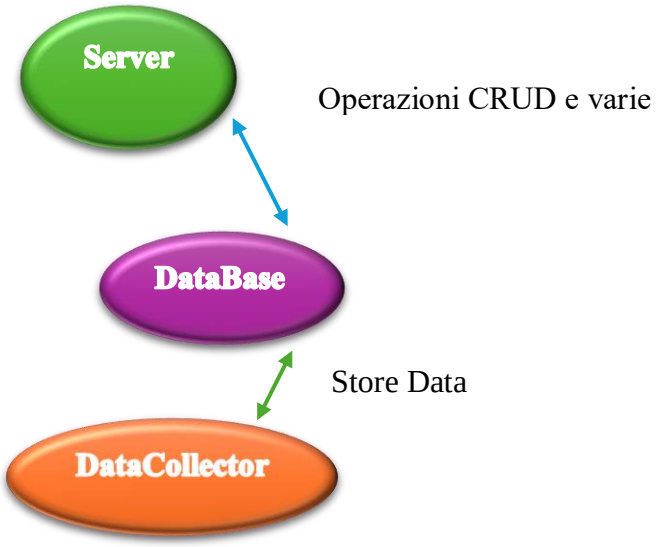

Figura 1: interazione microservizi

III. Interazioni tra i componenti

- **Client → Server:**

  Il client comunica mandando richieste gRPC al Server. Nello specifico il client prepara la richiesta inviando i parametri necessari e sulla base delle operazioni che vuole effettuare, stabilendo un canale grpc. Inoltre, il client permette di visualizzare i dati nel corretto formato che saranno restituiti dalla richiesta effettuata precedentemente, emettendo successive operazioni tramite interfaccia sui dati in possesso.

- **Server → Client:**

  Il server appena riceve e verifica la richiesta, elabora e genera i dati necessari. Successivamente il server restituisce al client una risposta contenente i dati elaborati e un messaggio che indica il corretto funzionamento dell'operazione. La risposta viene mandata sempre via canale gRpc precedentemente stabilito.

- **Server → Database:**

  Il server comunica direttamente con il database effettuando operazioni CRUD.
  - **Create**: effettuando la chiamata dell'API RegisterUser il server va ad inserire i nuovi utenti nella tabella con rispettivo Tickere con AddTicker va ad associare un nuovo Ticker ad un utente ed eventualmente aggiungerlo se non presente.
  - **Read**: Attraverso le chiamate Login, ShowTickersUser, GetLatestValue e GetAverageValue, va ad eseguire delle azioni di recupero dei dati dal database
  - **Update**: Nell'API di UpdateUser, permette di modificare i ticker associati ad un determinato utente.
  - **Delete**: tramite DeleteUser, DeleteTickerUser permettono al server di eliminare nelle rispettive tabelle e grazie alla relazione in cascata anche nelle altre associate.

  Inoltre, svolge operazioni di aggregazione utilizzato nel calcolo della media e di controllo come verifica la presenza di dati all'interno del databse

- **Database →Server**

Il database restituisce al server i dati richiesti tramite le query e permettere così poi al server di rispondere all'utente.

- **Database → Data Collector:**

Il database fornisce i ticker presenti che vengono utilizzati successivamente dal data collector per capire quali dati aggiornare e recuperare.

- **Data Collector → Database:**

  Il Data Collector responsabile di mantenere aggiornati i dati associati ai tickers comunica direttamente con il database. In specifico:
  - Recupera l'elenco dei ticker dal database.

Aggiorna valori associati ai ticker in maniera periodica.

- **Data Collector → Circuit Breaker**

Il DataCollector per controllare e gestire le chiamate verso le risorse utilizza il circuit breaker, consentendo al sistema di rilevare errori, timeout e permettendo il funzionamento anche quando le risorse non sono disponibili

- **Da Circuit Breaker → Data Collector:**

Il circuit breaker si comporta come una sorta di controllore; infatti, caso di errore o stato aperto il circuit breaker interrompe il flusso.

- **Circuit Breaker → Yahoo Finance**

Il circuit breaker nei confronti di yahoo finance si comporta come una sorta di meccanismo di protezione per le richieste e il recupero dei dati. Permette:

- **Protezione**: Infatti quando viene invocata la richiesta per accedere a yahoo finance il circuit breaker monitora le richieste rilevando errori timeout.
- **Valutazione flusso**: Viene eseguita una valutazione del flusso nei seguenti stati:
  - Stato Closed: se non sono stati rilevati errori il circuit breaker consente al Data Collector di inoltrare le richieste e il risultato torna indietro al Data Collector.
  - Stato Open: se vengono rilevati un numero elevato di errori consecutivi, apre il circuito e blocca le richieste del Data Collector evitando richieste verso la risorsa e viene segnalato tramite un messaggio di errore, permettendo di evitare ulteriori fallimenti o timeout su servizi esterni.
  - Stato Half-Open: Dopo un periodo di timeout, il circuit breaker entra in questo stato che permette di testare la risorsa ed effettuare un numero limitato di richieste. Nel caso in cui le richieste abbiano successo il circuit breaker torna nello stato "Closed" altrimenti torna nello stato "Open".

- **Da Yahoo Finance→Circuit Breaker:**

Yahoo Finance, quindi, fornisce le informazioni e i dati necessari. In caso di errori vengono gestiti dal circuit breaker.

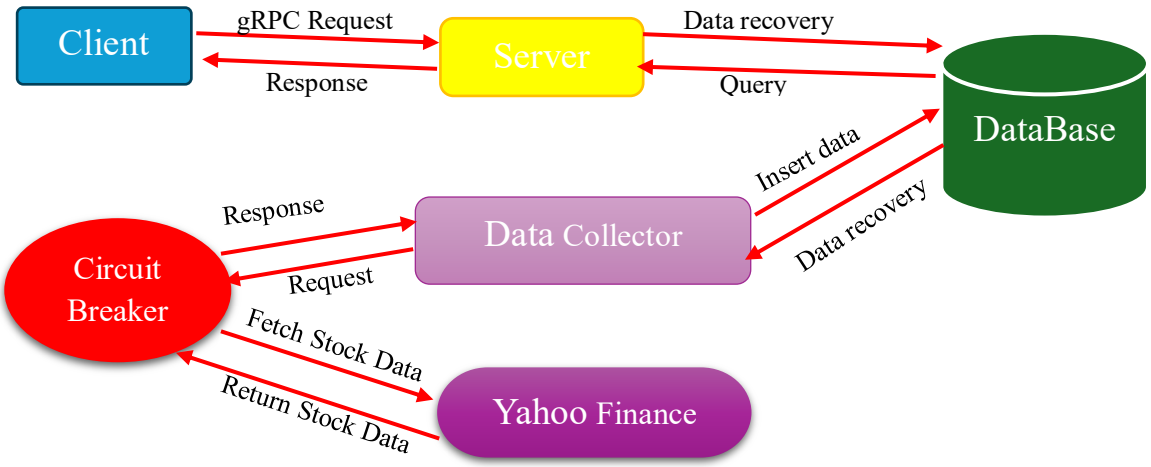

Figura 2: Interazioni componenti

IV. API Implementate

- **RegisterUser**: usata per registrare un nuovo utente, fornendo la propria e-mail e un ticker. Il sistema risponde alla richiesta con un messaggio e un valore boolean che mostra se la registrazione è avvenuta con successo o si è verificato un errore.
- **LoginUser**: permette all'utente già registrato di effettuare il login al sistema, inviando l'e-mail. Dopo che il sistema ha verificato l'effettiva presenza dell'e-mail nel database, viene mandato un messaggio e valore boolean che mostra se la richiesta è avvenuta con successo o si è verificato un errore.
- **AddTickerUtente**: permette all'utente di aggiungere un nuovo ticker, viene fornito il nuovo ticker, viene verificato che sia valido(presente nel file csv) e viene inviato al server tramite la richiesta che poi inserirà nel database.
- **ShowTickersUser**: permette di visualizzare tutti i ticker associati all'utente, non è necessario fornire manualmente l'e-mail dato il login dell'utente, Il sistema risponde alla richiesta con i tutti i ticker dell'utente che verranno mostrati e con un messaggio, un valore boolean che mostra se la modifica è avvenuta con successo o si è verificato un errore.
- **UpdateUser**: permette all'utente di aggiornare il proprio ticker, non è necessario fornire manualmente l'e-mail dato il login dell'utente. Il sistema risponde alla richiesta con un messaggio e un valore boolean che mostra se la modifica è avvenuta con successo o si è verificato un errore.
- **DeleteTickerUser:** permette di eliminare un ticker associato all'utente. Viene inserito un ticker e viene mandato al server che eliminerà il ticker associato a quell'utente e se non associati ad altri utenti verrà eliminato anche dal sistema.
- **GetLatestValue**: Restituisce il valore più recente del ticker inserito. Non è necessario mandare l'e-mail perché il sistema è già a conoscenza grazie al login effettuato in precedenza. Il sistema risponde alla richiesta con l'ultimo valore del ticker insieme al suo timestamp e restituendo un messaggio e un valore boolean che mostra se l'operazione è avvenuta con successo o si è verificato un errore.
- **GetAverageValue**: Calcola e restituisce il valore medio del ticker inserito. Non è necessario mandare l'e-mail perché il sistema è già a conoscenza grazie al login effettuato in precedenza, ma viene chiesto di quanti valori effettuare la media. Il sistema risponde alla richiesta con la media dei valori, il suo timestamp e restituendo un messaggio e un valore boolean che mostra se l'operazione è avvenuta con successo o si è verificato un errore.
- **DeleteUser**: permette di eliminare l'utente ed eliminare i ticker che sono associati esclusivamente ad esso e non ad altri user. Non è necessario inviare alcun dato, poiché il sistema successivamente al login è già a conoscenza della e-mail. Il sistema risponde alla richiesta con un messaggio e un valore boolean che mostra se l'eliminazione è avvenuta con successo o si è verificato un errore.

API	Descrizione	Input	Output
RegisterUser	Registra un utente e inserisci un ticker	E-mail, ticker	Messaggio, valore boolean in caso di conferma o errore
LoginUser	Login utente	E-mail	Messaggio, valore boolean in caso di conferma o errore
AddTickerUtente	Aggiunge ticker associato all'utente	Nuovo ticker	Messaggio, valore boolean in caso di conferma o errore
ShowTickersUser	Mostra i ticker associato all'utente	Non viene mandato nulla perché è implementata la funzione login.	valore boolean in caso di conferma o errore. Ritorna tutti i ticker associati all'utente.
UpdateUser	Aggiorna un ticker	Vecchio ticker, Nuovo ticker	Messaggio, valore boolean in caso di conferma o errore
DeleteTickerUser	Elimina un ticker associato all'utente	Ticker	Messaggio, valore boolean in caso di conferma o errore
GetLatestValue	Restituisce l'ultimo valore per un ticker	Ticker	Messaggio, valore boolean in caso di conferma o errore. Il ticker, ultimo valore e timestamp
GetAverageValue	Restituisce la media dei valori per un ticker	Ticker e numero che indica la media di quanti valori si vuole fare	Messaggio, valore boolean in caso di conferma o errore, Il ticker, la media e il timestamp
DeleteUser	Elimina un utente e i dati associati	Non viene mandato nulla perché è implementato la funzione login e-mail.	Messaggio, valore boolean in caso di conferma o errore

Tabella 1 Mostra le API implementate

V. Struttura Database

Si divide in quattro tabelle: Users, che gestisce gli utenti tramite e-mail, Tickers, gestisce i codici univoci per i ticker finanziari, UserTickers, che associa gli utenti ai ticker d'interesse e TickerData permette di memorizzare i dati storici relative a ticker.

– Users

Contiene gli utenti registrati nel sistema e vengono identificati in maniere univoca tramite il campo e-mail che funge da chiave primaria

Nome Campo	Tipo di Dato	Chiave	Relazione	Descrizione
e-mail	VARCHAR**(255)**	Priamary key		Indirizzo e-mail univoco che identifica l'utente

Tabella 2 USERS

– Tickers

Contiene i ticker associati a titoli finanziari, funge da chiave primaria e viene identificato in maniera univoca

Nome	Tipo di Dato	Chiave	Relazione	Descrizione
ticker	VARCHAR(10)	Primary key		Identificativo univoco

Tabella 3 Tickers

– UserTickers

Permette di stabile la relazione tra utenti e ticker, la combinazione di user e ticker costituisce la chiave primaria, inoltre entrambi si collegano come chiave esterna alla tabella user e ticker

garantendo l'autoeliminazione nel caso viene eliminato un utente o un ticker grazie al comando ON DELETE CASCADE

Nome	Tipo di Dato	Chiave	Relazione	Descrizione
user	VARCHAR(255)	Primary key	Collegato a **Users.email** (ON DELETE CASCADE)	E-mail dell'utente che possiede il ticker.
ticker	VARCHAR(10)	Primary key	Collegato a **Tickers.ticker** (ON DELETE CASCADE)	E-mail dell'utente che possiede il ticker.

Tabella 4 UserTickers

- **TickerData**

Contiene i dati storici relative ai ticker. Il campo id viene utilizzato come chiave primaria ed è univoco, generato automaticamente. La colonna ticker chiave esterna collega alla tabella tickers. La colonna value memorizza il valore numerico del ticker mentre la colonna timestamp tiene conto della data e l'ora in cui viene registrato. Tramite il comando ON DELETE CASCADE ticker permette l'autoeliminazione nel caso in cui non siano collegato a nessun ticker.

Nome	Tipo di Dato	Chiave	Relazione	Descrizione
id	INT AUTO_INCREMENT	Primary Key		Identificativo univoco
ticker	VARCHAR(10)	Foreign Key	Collegato a **Tickers.ticker** (ON DELETE CASCADE)	Identificativo del ticker
value	INT	Foreign Key		Valore associato al ticker.
timestamp	STRING			Data e ora in cui è stato registrato il valore.

Tabella 5 TickerData

Documento redatto da:

Massimiliano Finocchiaro, Dario Rovito